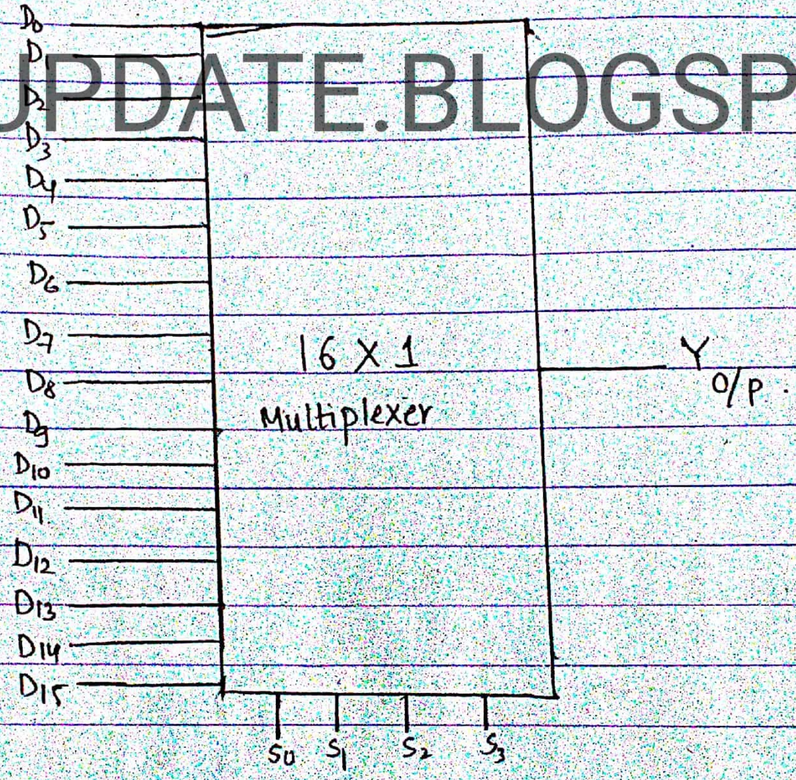
– Gaurav Chaulagain
– Digital Logic

2065

1. Draw a block diagram, truth table and logical circuit of a 16 X 1 multiplexer and explain its working principle.

→

Block diagram:



Working principle:

A multiplexer is a combinational circuit that allows digita information from several sources to be routed on a single line of transmi It accepts data from any input sources for transmission over a common shared line.

A MUX has several data input lines and a single output lin and selection switches which permits digital data on any one of the inputs to be switched to the output lines. A MUX has 2^n inputs and 'n' select bits.

for 16 x 1 MUX,

datat inputs = $2^4 = 16$

data output = 1.

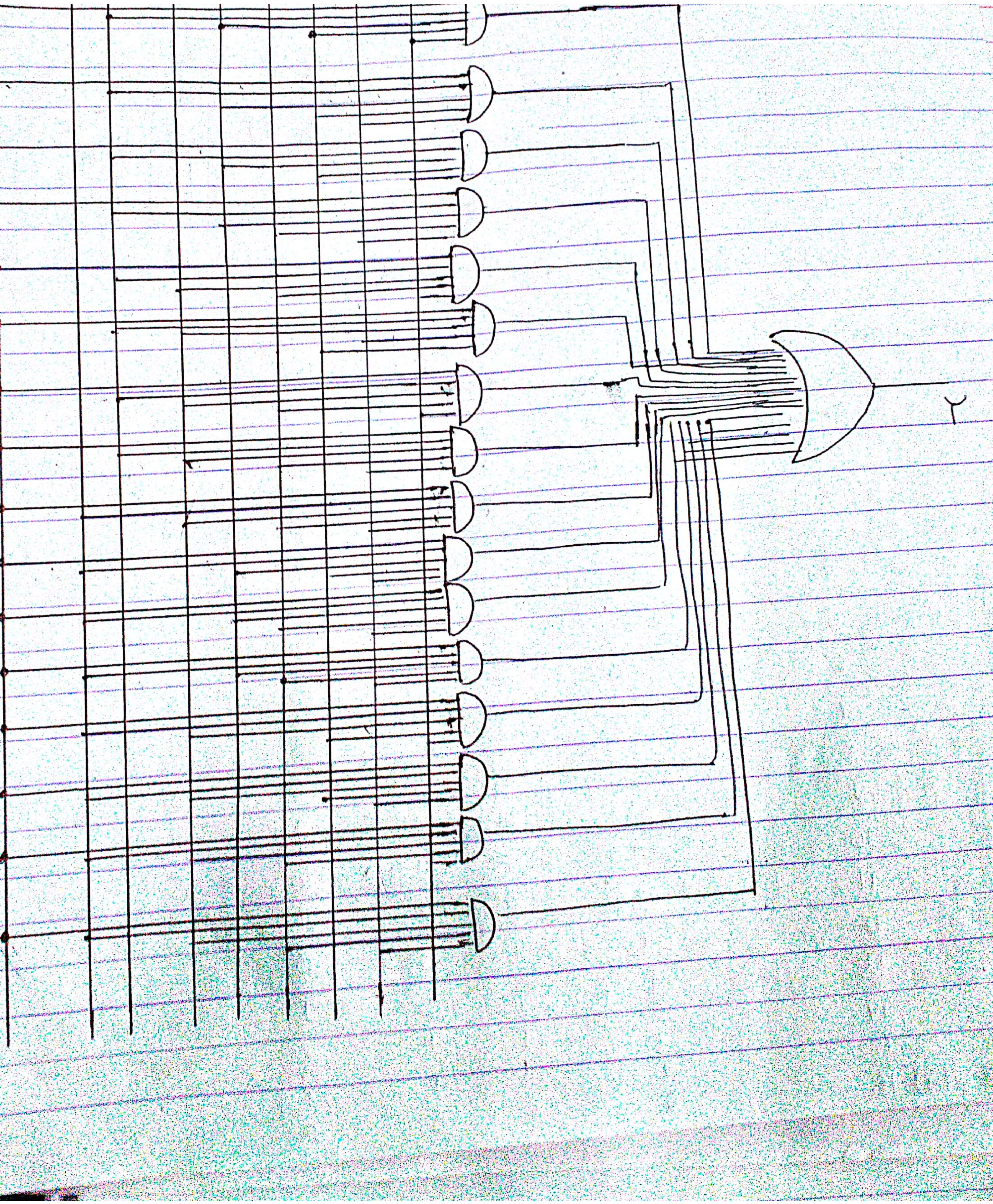
select bits = 4.

Truth Table:

Data I/P	S_0	S_1	S_2	S_3	Y
D_0	0	0	0	0	$D_0 \bar{S_0} \bar{S_1} \bar{S_2} \bar{S_3} \Rightarrow D_0$ selected
D_1	0	0	0	1	$D_1 \bar{S_0} \bar{S_1} \bar{S_2} S_3 \Rightarrow D_1$ selected
D_2	0	0	1	0	$D_2 \bar{S_0} \bar{S_1} S_2 \bar{S_3} \Rightarrow D_2$ selected
D_3	0	0	1	1	$D_3 \bar{S_0} \bar{S_1} S_2 S_3 \Rightarrow D_3$ selected
D_4	0	1	0	0	$D_4 \bar{S_0} \bar{S_1} \bar{S_2} \bar{S_3} \Rightarrow D_4$ selected
D_5	0	1	0	1	$D_5 \bar{S_0} S_1 \bar{S_2} S_3 \Rightarrow D_5$ selected
D_6	0	1	1	0	$D_6 \bar{S_0} S_1 S_2 \bar{S_3} \Rightarrow D_6$ selected
D_7	0	1	1	1	$D_7 \bar{S_0} S_1 S_2 S_3 \Rightarrow D_7$ selected
D_8	1	0	0	0	$D_8 S_0 \bar{S_1} \bar{S_2} \bar{S_3} \Rightarrow D_8$ selected
D_9	1	0	0	1	$D_9 S_0 \bar{S_1} \bar{S_2} S_3 \Rightarrow D_9$ selected
D_{10}	1	0	1	0	$D_{10} S_0 \bar{S_1} S_2 \bar{S_3} \Rightarrow D_{10}$ selected
D_{11}	1	0	1	1	$D_{11} S_0 \bar{S_1} S_2 S_3 \Rightarrow D_{11}$ selected
D_{12}	1	1	0	0	$D_{12} S_0 S_1 \bar{S_2} \bar{S_3} \Rightarrow D_{12}$ selecte
D_{13}	1	1	0	1	$D_{13} S_0 S_1 \bar{S_2} S_3 \Rightarrow D_{13}$ selecte
D_{14}	1	1	1	0	$D_{14} S_0 S_1 S_2 \bar{S_3} \Rightarrow D_{14}$ selecte
D_{15}	1	1	1	1	$D_{15} S_0 S_1 S_2 S_3 \Rightarrow D_{15}$ select

logic circuit:

$\Rightarrow$


Y

2. Explain the 4-bit ripple counter and also draw a timing diagram.

→

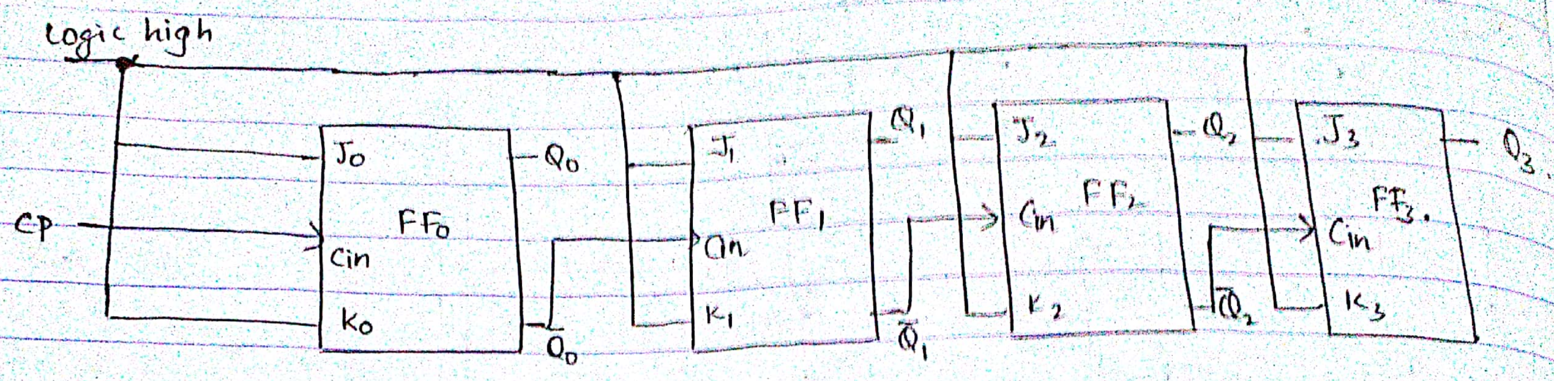


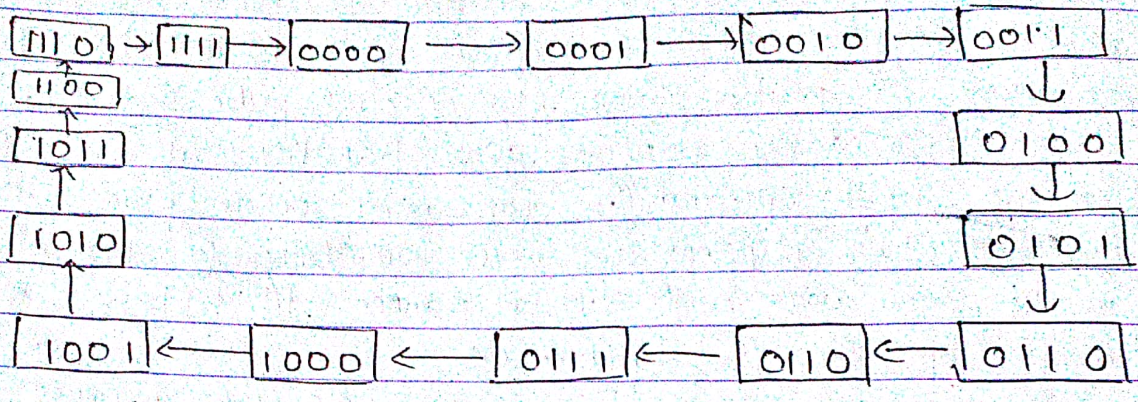
fig. 4-bit ripple Counter

state sequence.

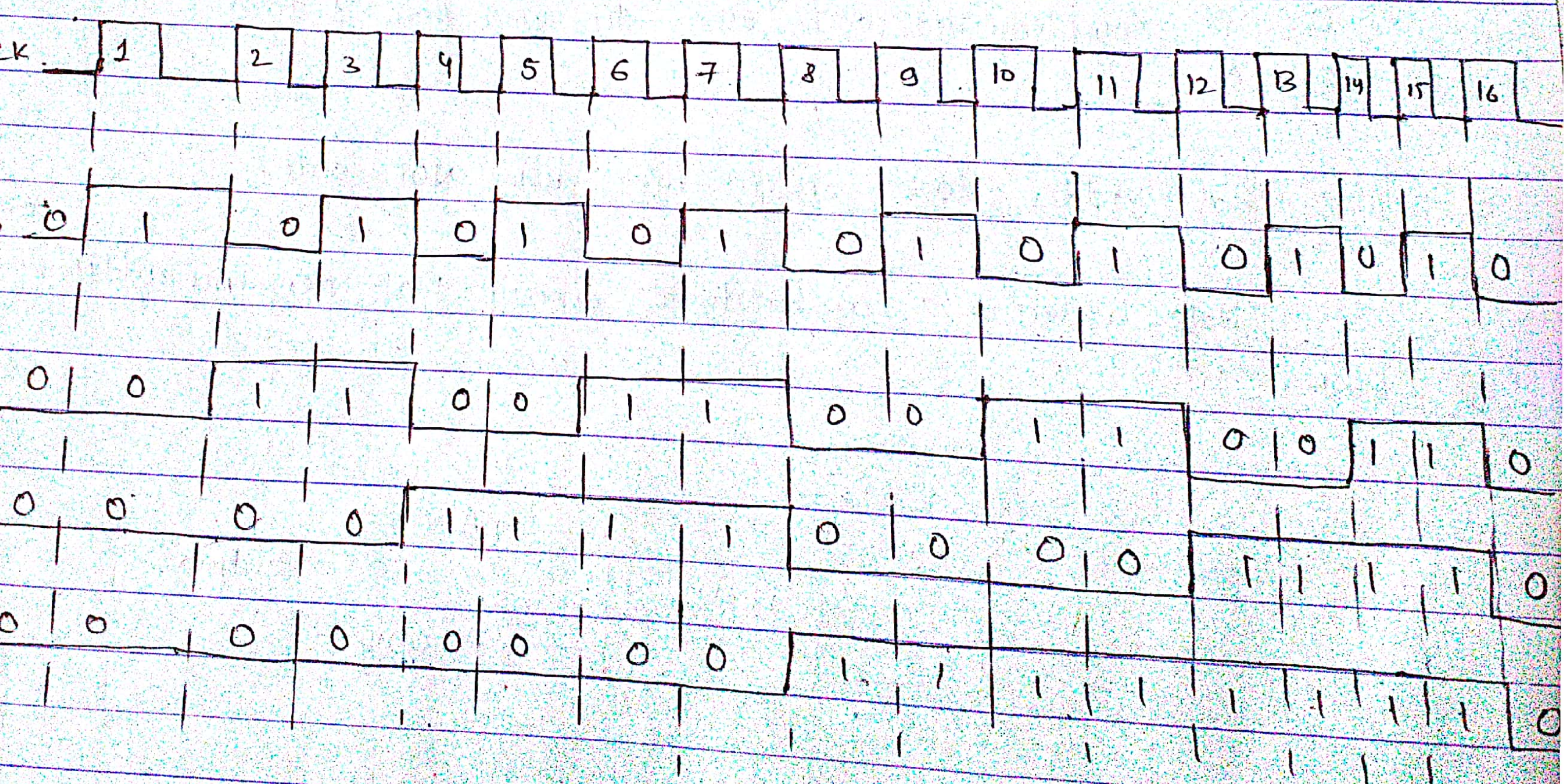
clock pulse	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
(Recycler) 16	0	0	0	0

Scanned by CamScanner

state diagram:



iming diagram:



Here, 4 flipflops are connected with CP applied to C input of the first flipflop (FF0) which is always the least significant bit. The second flipflop (FF_1) is triggered by $\overline{Q}_0$ [output of FF_0]. FF_0 changes state at positive going edge of clock pulse but FF_1 changes only when triggered by positive going transition of $\overline{Q}_0$. Since, transition of input Cp and $\overline{Q}_0$ can never occur exactly at the same time, the two flipflops are never simultaneously triggered. And likewise, in the same way with similar manner FF_2, FF_3 are triggered and the counter operation is asynchronous.

The 4-bit ripple counter has 16 states because of 4-flip flops. This is an up-counter which counts from 0 to 15. where,

- Q_0 is complemented at every count pulse
- Q_1 is complemented only when Q_0 goes from 1 to 0.
- Q_2 is complemented only when Q_1 goes from 1 to 0.

~~and Q_3~~ Q_3

. Design Half adder logic circuit only using NOR gate.

→

Half adder is a combinational circuit that performs addition of 2 binary bits.

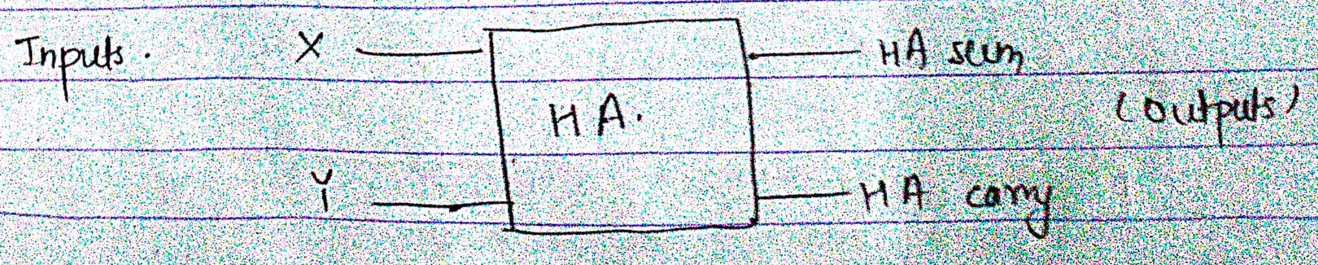


fig. Block diagram

Scanned by CamScanner

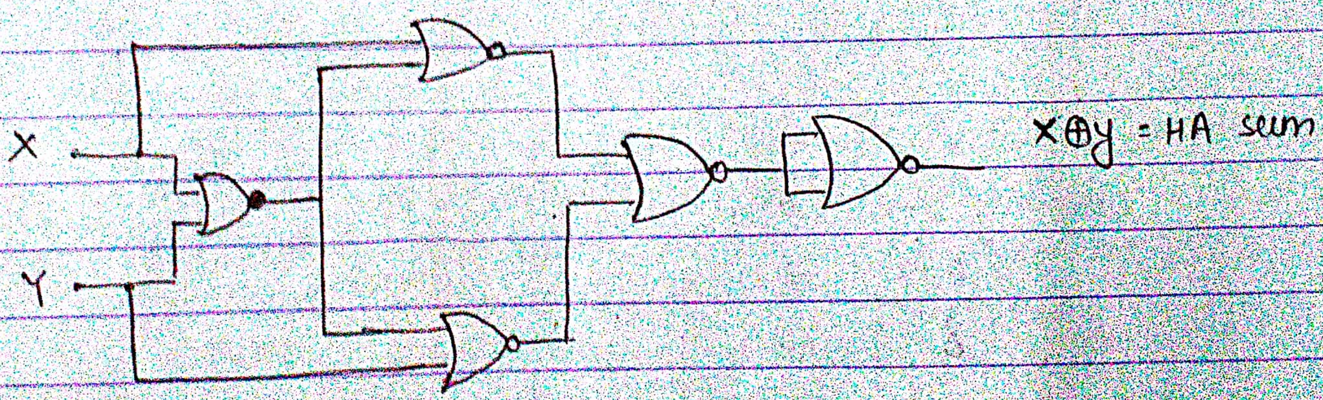
Truth table:

Inputs		Outputs	
X	Y	HA sum	HA carry
0	0	0	0
1	0	$\boxed{1}$	0
0	1	$\boxed{1}$	0
1	1	0	$\boxed{1}$

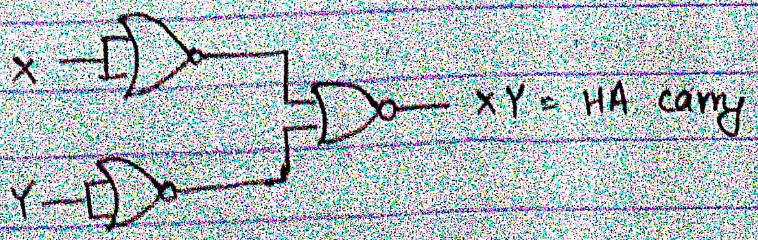
$$HA \ sum = xy' + x'y = x \oplus y$$
$$HA \ carry = xy$$

Only using Nor gates, [logic circuit].

HA sum:



HA carry:



5. Convert the following decimal numbers into hexadecimal and Octal number.

(a) 304

Converting into octal:

$$\begin{array}{r|l} 8 & 304 \rightarrow 0 \\ 8 & 38 \rightarrow 6 \\ & 4 \rightarrow 4 \end{array} \uparrow$$

$\therefore (304)_{10} = (460)_8$.

Converting into hexadecimal:

$$\begin{array}{r|l} 16 & 304 \rightarrow 0 \\ 16 & 19 \rightarrow 3 \\ & 1 \rightarrow 1 \end{array} \uparrow$$

$\therefore (304)_{10} = (130)_{16}$

(b) 224

Converting into octal:

$$\begin{array}{r|l} 8 & 224 \rightarrow 0 \\ 8 & 28 \rightarrow 4 \\ & 3 \rightarrow 3 \end{array} \uparrow$$

$\therefore (224)_{10} = (340)_8$

Converting into hexadecimal:

$$\begin{array}{r|l} 16 & 224 \rightarrow 0 \\ & 14 \rightarrow 14 \end{array} \uparrow$$

$\therefore (224)_{10} = (140)_{16}$

6) Describe the three variable k-map with example:

→ The karnaugh map or k-map is a graphical technique for simplifying boolean function. The k-map is two dimensional representation of truth table, It provides a simpler method for minimizing logic expressions. The map method is ideally suited for four or less variables. A k-map is diagram consisting of squares. Each square representing either min term or max term. Any logic expression can be written either in sum of product term or product of seem term.

A karnaugh map for n variables are made up of 2^n squares. Each square designate a product term in Boolean expression.

for three variable kanaugh-map, there are 3 variables i.e. the k-map consists of 8 squares. ~~Each square for product~~

for any boolean function,

$$F(W, X, Y) = \Sigma(0, 1, 3, 4)$$

k-map is written as,

W \ XY	X'Y'	X'Y	XY	XY'
W'	1	1	1	
W	1			

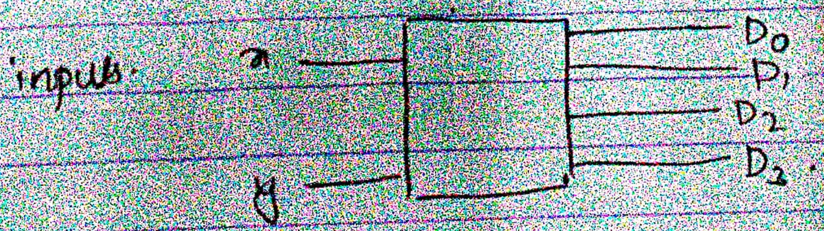
The reduced boolean expression is

$$F = X'Y' + Y$$

7) Design the Decoder using universal gates:

→

2x4 decoder:



Truth table:

inputs		outputs			
X	Y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Boolean expression:

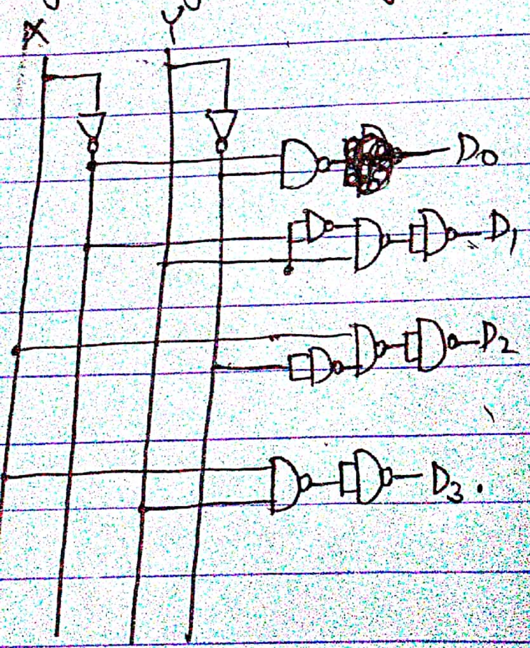
$D_0 = x'y'$

$D_1 = x'y$

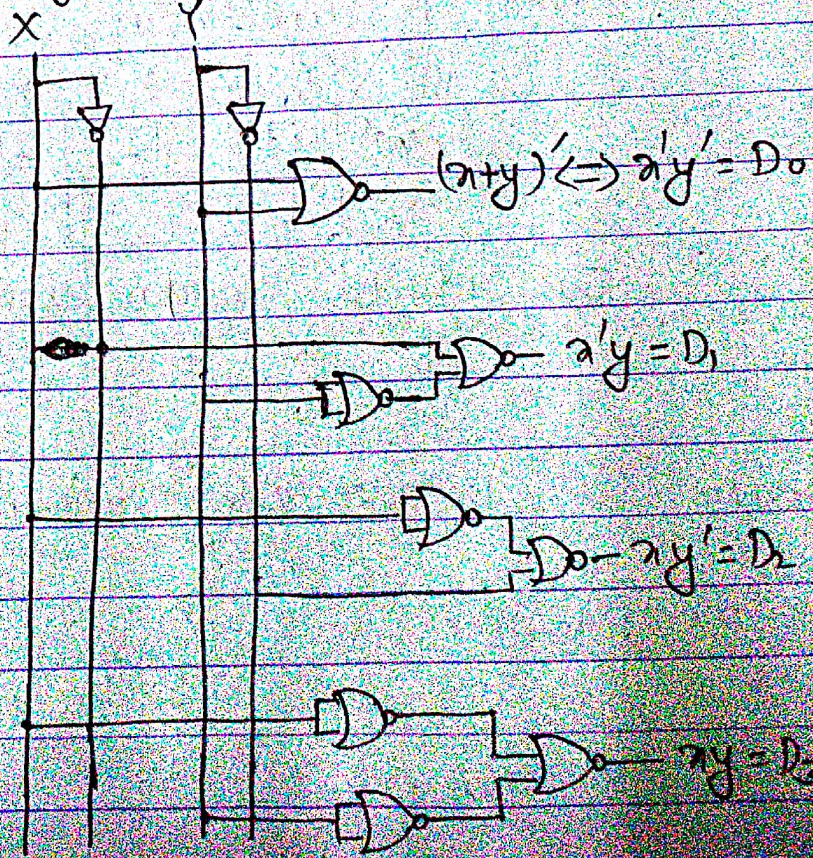
$D_2 = xy'$

$D_3 = xy.$

logic diagram using NAND gate

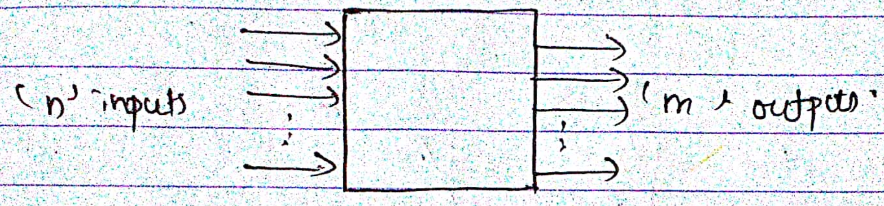


logic diagram using NOR



8: What is combinational circuit? What are its important features?

→ Combinational circuit consists of logic gates whose output at any time is determined by combining the values of the applied inputs using logic operators. Performs an operation that can be specified logically by a set of boolean expressions. It consists of input variables, output variables, logic gates and interconnections which accepts signals from the input and generates signals at the output. For 'n' input variables, there are '2^n' possible input combinations.



Characteristics:

i) The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.

ii) The combinational circuit does not use any memory. The previou state of input does not have any effect on present state of the output.

iii) A combinational circuits can have 'n' number of inputs and 'm' no. of outputs.

9. Describe the Clocked RS flipflop

→ A basic flipflop in an asynchronous sequential circuit which ha no clock pulse & by adding gates to the input of the basic circuits, the flipflop can be made to respond to the input levels during the occurance a clock pulse. A clocked RS flipflop consists of basic NOR flipflop and two NAND gates or basic NAND flipflop and two NAND gates.

The outputs of two AND gates remain at 0 as long as CP is 0 regardless of S and R input values. When CP goes 1, information from S and R inputs is allowed to reach the basic flip flop. The SET state is reached with S=1, R=0 and CP=1. CLEAR state is reached with S=0, R=1 and CP=1. If S=1 and R=1, the occurrence of clockpulse causes both outputs to momentarily goto 0. This is called indeterminate state and has to be practically avoided. If S=0 and R=0, the flipflop retains the previous state.

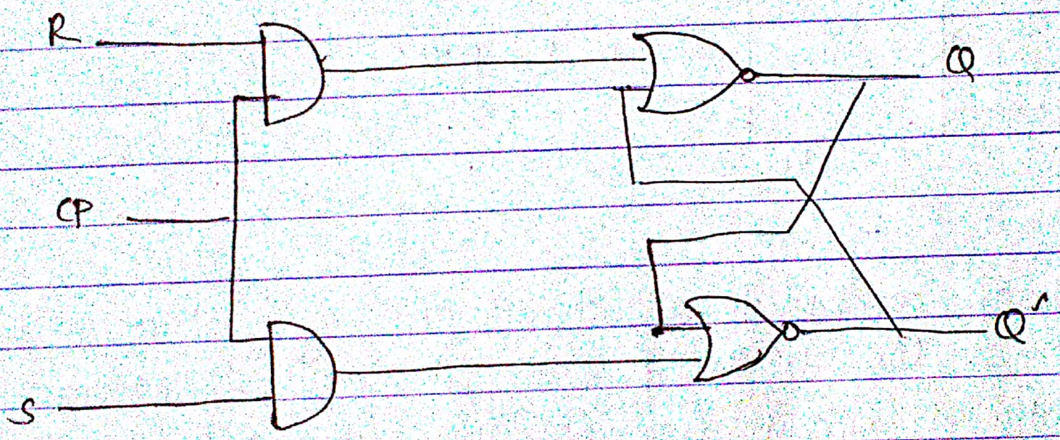


fig. clocked RS flip using basic flipflop and AND gate.

11. What are shift register operations?

→

Shift register is a register capable of shifting its binary information either to the right or to the left. The logical configuration of a shift register consists of a chain of flipflops connected in a cascade with the output of one flipflop connected to the input of another flip. All flipflops recieves on common clock pulse.

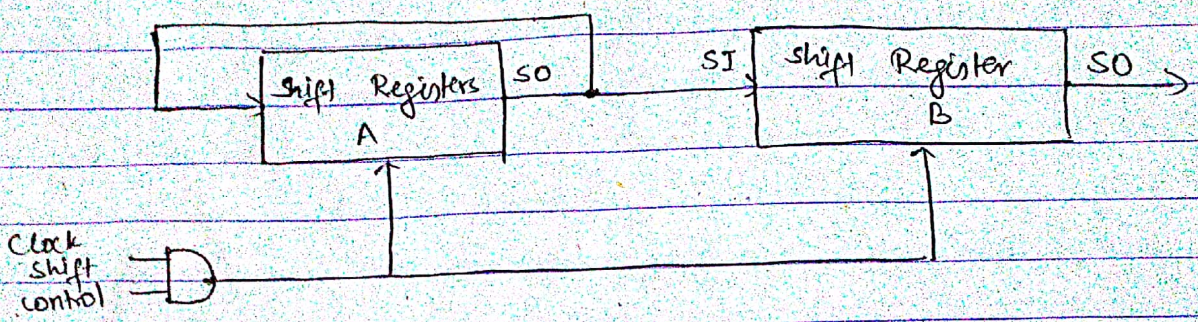


fig. Block diagram of serial transfer using shift registers.

The serial transfer of information from register-A to register-B is done usinging shift registers. The serial output, SO of register-A goes to the serial ~~output~~ input of register-'B'. To prevent the loss of information stored in the source register, the values of register-A is made to circulate by connecting the SO to its SI terminal. The initial cont of register-B is shifted out through the serial output and is lost unless it is transferred to a third shift register.

– Gaurav Chaulagain
– Digital Logic

2066

1. Design the 4-bit Synchronous up/down counter with timing diagram, logic diagram and truth table.

→ It is a counter that is capable of procesing in other direction through a certain sequence which is also called bi-directional counter.

A 4-bit counter that advances upward through its sequence 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 & then downwards ~~till~~ from 15 till 0.

Up/down counter can be reversed at any point in their sequence.

For example:

$\underbrace{0, 1, 2, 3, 4, 5}_{up}$, $\underbrace{4, 3, 2}_{down}$, $\underbrace{3, 4, 5, 6, 7}_{up}$, $\underbrace{6, 5, 4}_{down}$, $\underbrace{5, 6, 7}_{up}$.

Binary sequence:

C.P	UP	Q_3	Q_2	Q_1	Q_0	Down
0		0	0	0	0	
1		0	0	0	1	
2		0	0	1	0	
3		0	0	1	1	

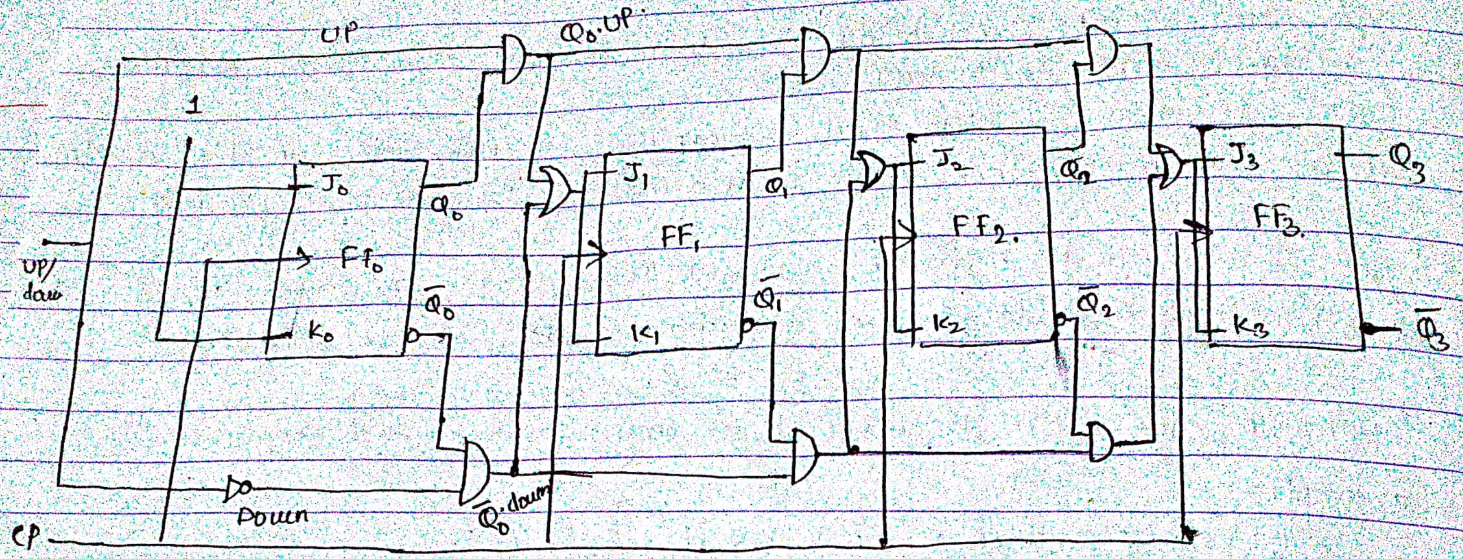


fig. logic diagram of UP/Down synchronous counter (4-bit)

Here,

i) Q_0 changes on its CP i.e. $J_0 = K_0 = 1$.

ii) for UP sequence, Q_1 changes state when $Q_0 = 1$ and for down sequence, Q_1 changes state when $Q_0 = 0$ i.e. $\boxed{J_1 = K_1 = Q_0 \cdot UP + Q_0' \cdot down}$

iii) For UP sequence, Q_2 changes state when $Q_0 = Q_1 = 1$ and for down sequence, Q_2 changes state when $Q_0 = Q_1 = 0$ i.e. $\boxed{J_2 = K_2 = Q_0 \cdot Q_1 \cdot UP + Q_0' \cdot Q_1' \cdot down}$

iv) For UP sequence, Q_3 changes state when $Q_0 = Q_1 = Q_2 = 1$ and for down sequence, Q_3 changes state when $Q_0 = Q_1 = Q_2 = 0$ i.e. $\boxed{J_3 = K_3 = Q_0 \cdot Q_1 \cdot Q_2 \cdot UP + Q_0' Q_1' \cdot Q_2' \cdot down}$

Scanned by CamScanner

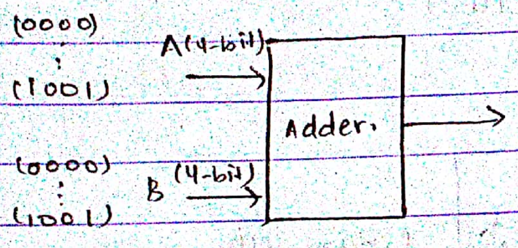
3. Design a decimal adder with logic diagram and truth table.

→

A BCD adder is a circuit that adds two BCD digits in parallel and produces a sum of digits also in BCD upto 9 i.e. 0 to 9. The corresponding BCD is identical to the binary, so no correction is required.

But, when the input binary sum is greater than 9 i.e. 1001, and the BCD sum is not equivalent to the binary sum. So, correction is required i.e. the addition of binary '6' [0110] ; to the binary sum converts it to the BCD representation and also produces required carry.

The correction can be derived from the truth table.



output of the Addition

(0 − 18).

max. value of $(A+B) = (9+9) = 18$.

min. value of $(A+B) = 0$

when, considering carry : $\boxed{(0-19)}$

Decimal	Binary sum C^*	S_3^*	S_2^*	S_1^*	S_0^*	BCD sum C	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

Analysing, the truth table, we can clearly see that the binary sum and BCD sum are not equivalent from 10 to 19. so, we have to apply correction i.e. adding "0110" when :

(i) $c^* = 1$

(ii) $S_3^* \cdot (S_2^* + S_1^*)$

(iii) $S_3^* \cdot S_1^*$

Now,

The condition for correction and output carry can be expressed by boolean formula,

$$C = c^* + S_3^* \cdot (S_2^* + S_1^*) + S_3^* \cdot S_1^*$$
$$= C^* + S_3^* \cdot S_2^* + S_3^* \cdot S_1^*$$

When, $C=1$, it is necessary to add 0110 to the binary sum :

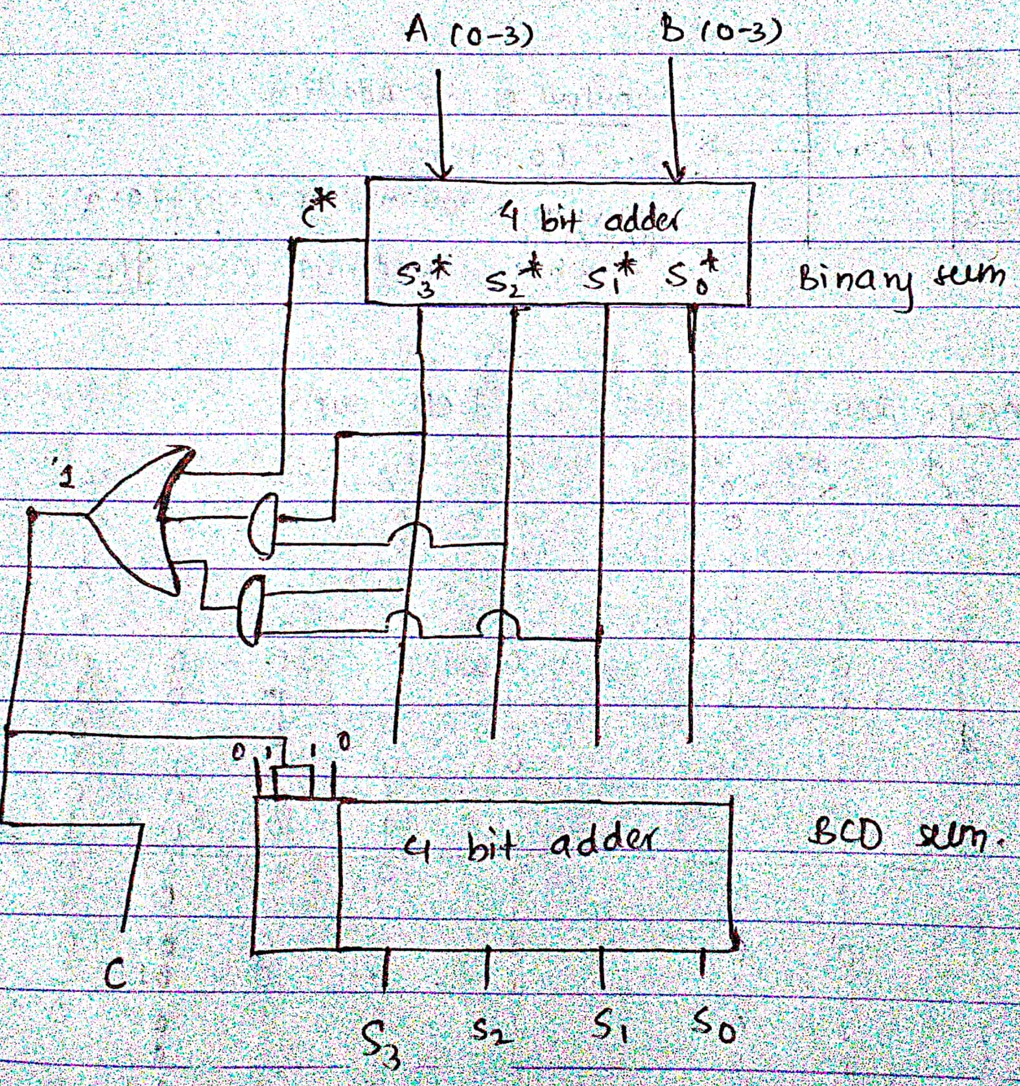


fig. logic diagram of BCD adder

5. Convert the following octal numbers to hexadecimal.

a) 1760.46.

solution:

$(1760.46)_8 = (?)_{16}$.

First:

changing octal into binary:

1	7	6	0	.	4	6.
001	111	110	000		100	110

$(1760.46)_8 = (111\,110\,000.\,100110)$.

Now,

changing binary into hexadecimal:

0001	1111	0000	.	1001	1000
1	F	0		9	8.

$\therefore (1760.46)_8 = (1F0.98)_{16}$.

b) 6055.263.

solution:

$(6055.263)_8 = (?)_{16}$.

· changing octal to binary:

6	0	5	5	.	2	6	3
110	000	101	101		010	110	011

$\therefore (6055.263)_8 = (110000101101.010110011)_2$

Now,

changing binary into hexadecimal:

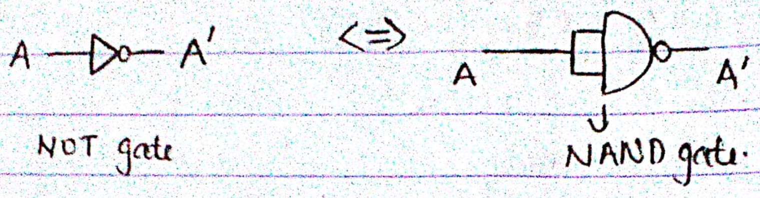
1100	0010	1101	.	0101	1001	1000
C	2	D		5	9	8

$\therefore (6055.263)_8 = (C2D.598)_{16}$

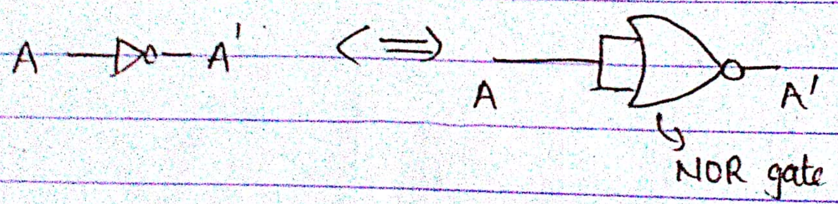
6. Which gates can be used as inverts in additional to the NOT gate and how?

→ NAND and NOR gates can be used as NOT gate. They can be made into NOT gate by manipulating the single input as numerous inputs as shown in the figure:

(i) NOT using NAND:



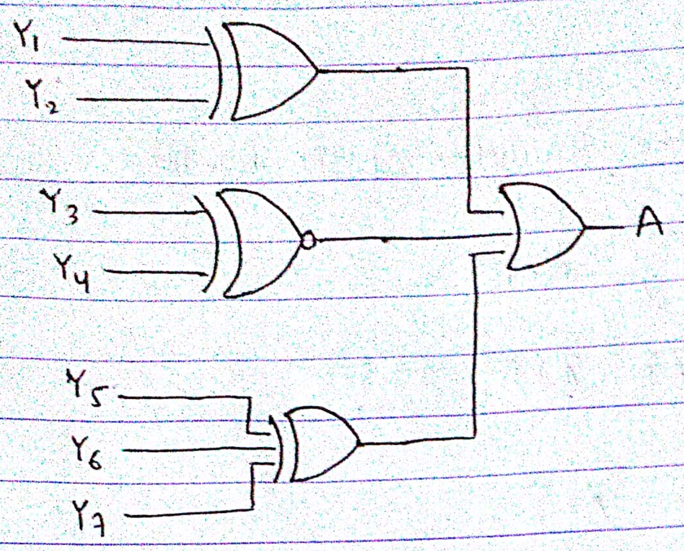
(ii) NOR as NOT:



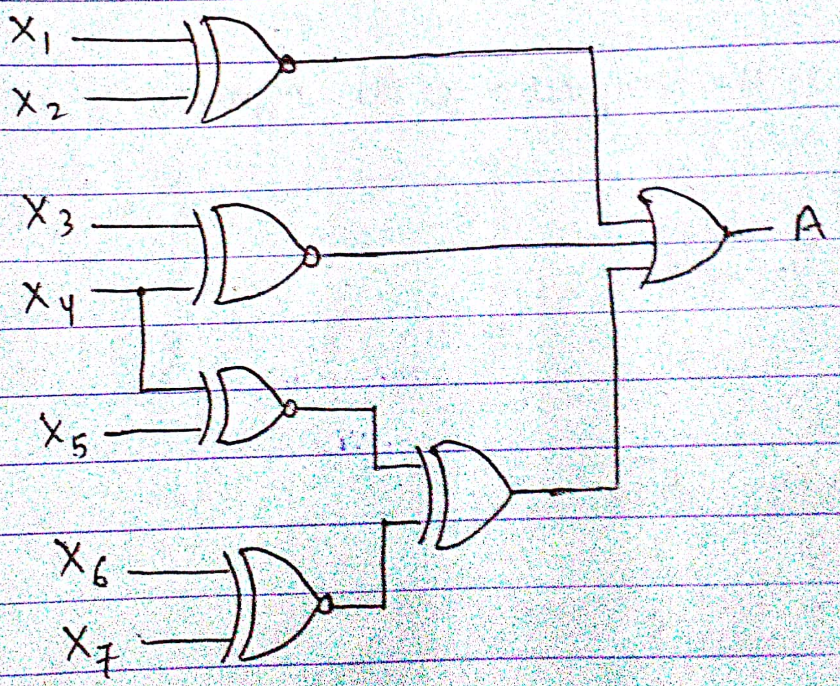
7. Draw a logic gates that implements the following:

a) $A = (Y_1 \oplus Y_2)(Y_3 \odot Y_4) + Y_5 \oplus Y_6 \oplus Y_7$.

⇒



b) $A = (X_1 \odot X_2) + (X_3 \odot X_4) + (X_4 \odot X_5) \oplus (X_6 \odot X_7)$



Scanned by CamScanner

8. State and prove De-Morgan's theorem 1st and 2nd with logic gates and truth table:

→ There are two De-Morgan's theorem. They are:

i) The negation of a conjugation is the disjunction of the negations.

$\overline{A \cdot B} = \overline{A} + \overline{B}$.

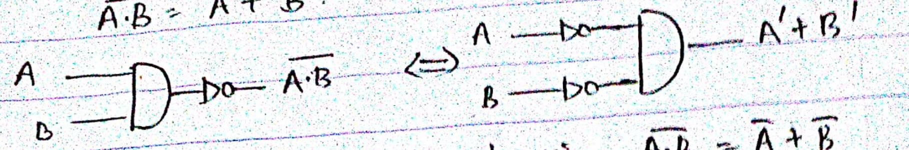


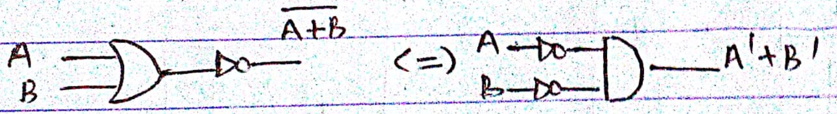
fig. logic gates showing $\overline{A \cdot B} = \overline{A} + \overline{B}$

Truth table:

A	B	$\overline{A}$	$\overline{B}$	$A \cdot B$	$\overline{A \cdot B}$	$\overline{A} + \overline{B}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

ii) The negation of a disjunction is the conjunction of the negations.

$\overline{A + B} = \overline{A} \cdot \overline{B}$.



Truth table:

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$	$A + B$	$\overline{A + B}$
0	0	1	1	1	0	1
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	1	0	0	0	1	0

9. Reduce the following expression using K-map.

$$F = \bar{A} + B(A+\bar{B}+D)(\bar{B}+C)(B+C+D)$$
$$= \bar{A} + (AB+BD)(\bar{B}+C)(B+C+D)$$
$$= \bar{A} + (AB+BD)(\bar{B}C+\bar{B}D+BC+C+CD)$$
$$= \bar{A} + \cancel{AB\bar{B}C}^{\,0} + \cancel{AB\bar{B}D}^{\,0} + ABBC + ABC + ABCD + \cancel{B\bar{B}CD}^{\,0} + \cancel{B\bar{B}DD}^{\,0} +$$
$$BBCD + BCD + BCDD$$

$$= \bar{A} + ABC + ABC + ABCD + BCD + BCD + BCD$$
$$= \bar{A} + ABC + BCD + ABCD$$
$$= A'B'C'D' + \underline{A'BCD} + ABCD' + \underline{ABCD} + \underline{ABCD} + \underline{A'BCD} + \underline{ABCD}$$
$$= A'B'C'D' + A'BCD + ABCD' + ABCD$$

using K-Map,

	$C'D'$	$C'D$	CD	CD'
$A'B'$	1			
$A'B$			1	
AB			1	1
AB'				

$$F = A'B'C'D' + BCD + ABC.$$

~~11. Explain the operation of Decoder~~:

10. Difference between a MUX and DEMUX

→

Multiplexer (MUX):

i) Many inputs & one output

ii) Data select lines

iii) Parallel to serial conversion

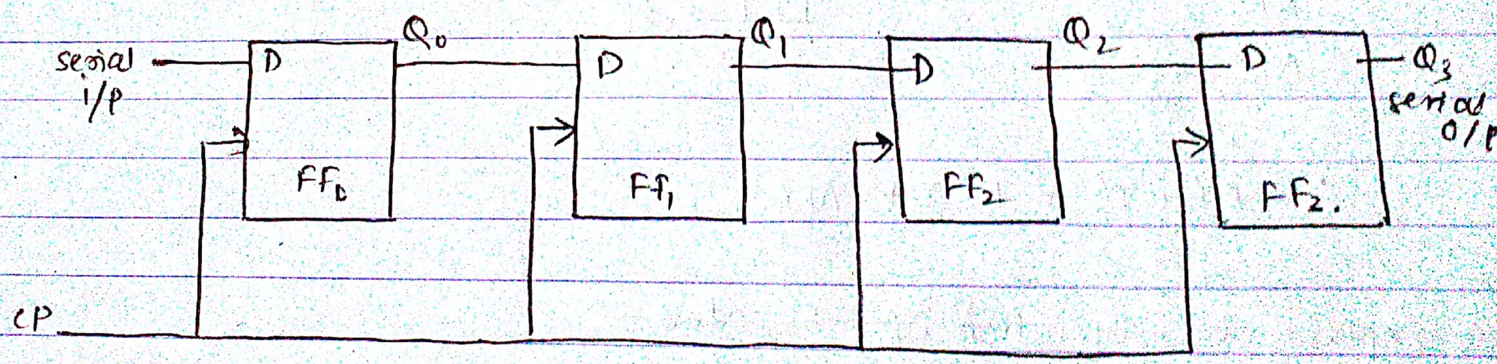
iv) When we design MUX, we don't need additional gates.

v) Example: 8:1, 16:1, 32:1.

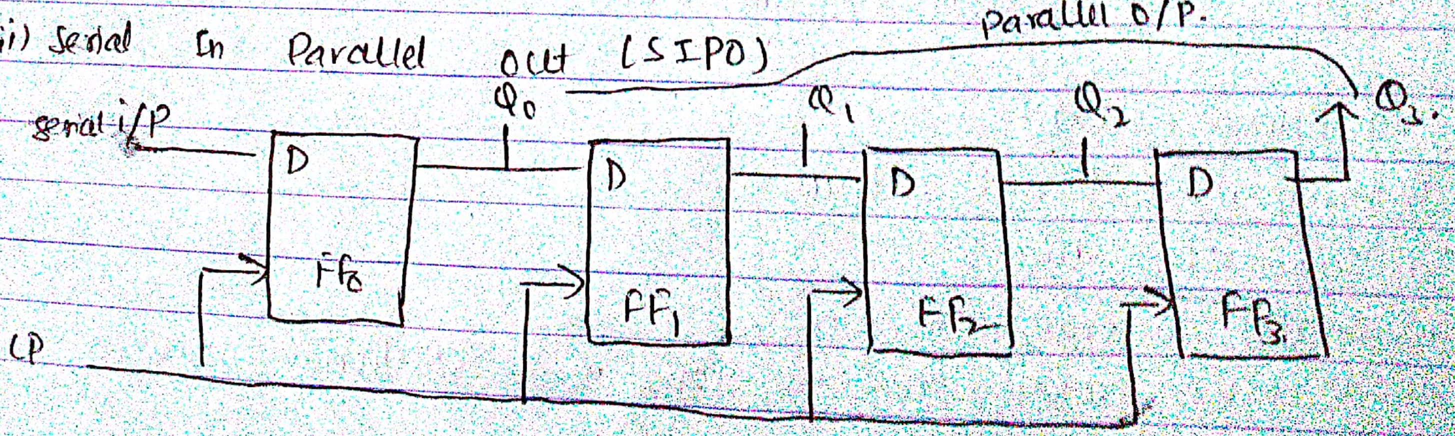
Demultiplexer:-
i) one inputs and many outputs
ii) Data distributer
iii) serial to parallel conversion
iv) When we design demultiplexer, we need additional gates.
v) Example: 1:8, 1:16, 1:32.

12. What are the various types of shift registers:
→ There are four types of shift registers. They are:

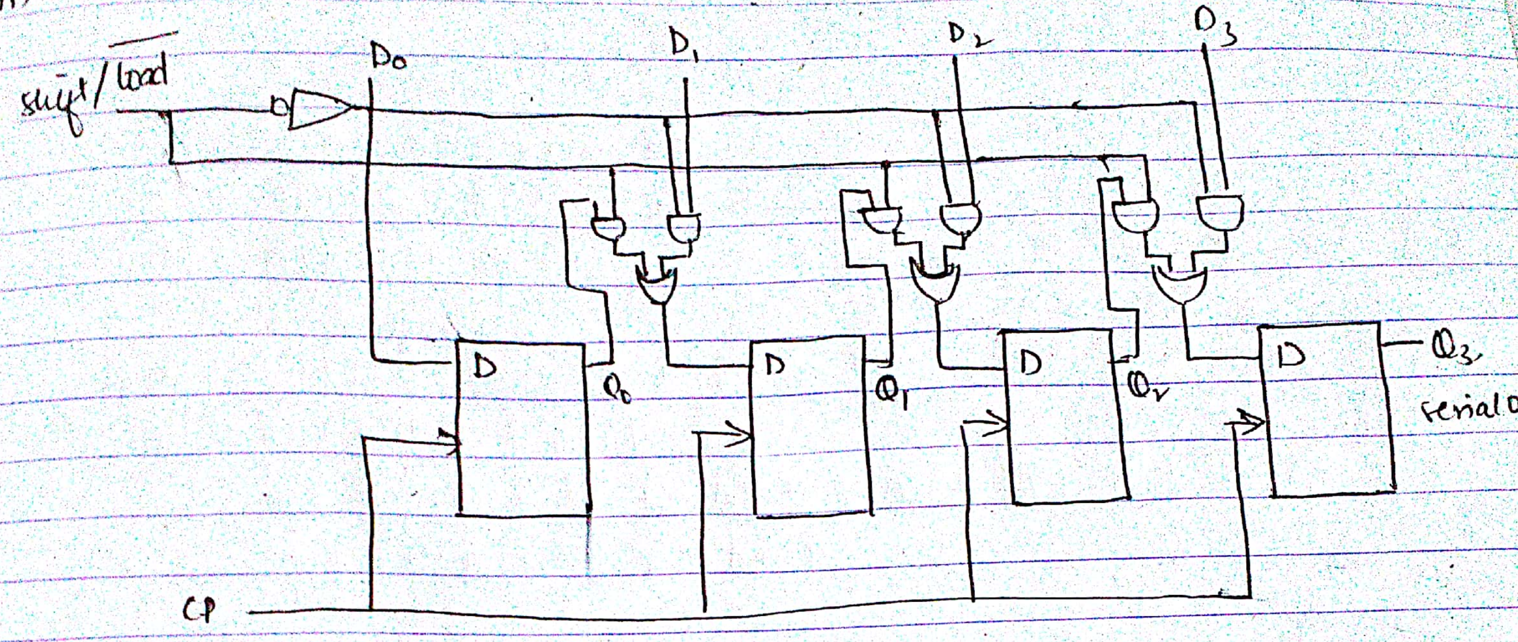
(i) Serial In Serial out (SISO)



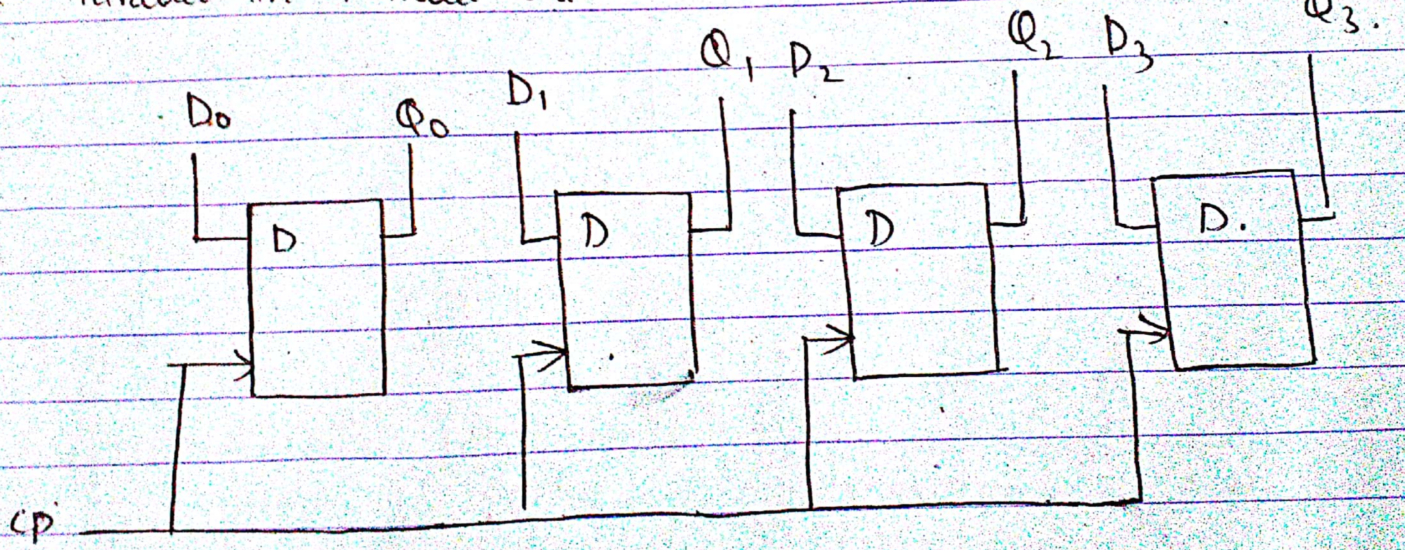
(ii) Serial In Parallel out (SIPO)



(iii) Parallel In Serial Out



(iv) Parallel In Parallel Out:



- Gaurav Chaulagain
- Digital Logic

2070

1. Explain magnitude comparator and also design a logic diagram for a 4-bit magnitude comparator.

→

A Magnitude comparator is a digital comparator which has three output terminals, one for each : equality ; A = B , greater than ; A > B , and less tha A < B.

The purpose of digital comparator is to compare a set of variabl are unknown numbers. for example ; a magnitude comparator of two '1-bits, (A and B) inputs would produce the following three output conditions w compared to each other.

A > B , A = B , A < B .

This is useful if we want to compare two variables and want to produce an output when any of the above three conditions are achieved.

4-bit magnitude comparator:

Since, A = 4 bits

B = 4 bits

let , $A = A_3 \quad A_2 \quad A_1 \quad A_0$

$B = B_3 \quad B_2 \quad B_1 \quad B_0$

Firstly, we try to verify equality comparison.

for that,

let us suppose , $X_i = A_i \odot B_i = A_i B_i + A_i' B_i'$

⇒

In order to verify equality comparison, X_i should be equal to 1 i.e. $X_i = 1$.

But, $X_i = 1$ if and only if $A_i = B_i$ for all $i = 0, 1, 2, 3$. Therefore, the condition for $A = B$ or Equality (EQ) = 1 if and only if:

$A_3 = B_3 \Rightarrow (X_3 = 1)$, and

$A_2 = B_2 \Rightarrow (X_2 = 1)$, and

$A_1 = B_1 \Rightarrow (X_1 = 1)$, and

$A_0 = B_0 \Rightarrow (X_0 = 1)$

Then the boolean expression for equality to be equal to '1' will be

$$EQ = X_3 X_2 X_1 X_0 \quad \text{——— (i)}$$

Then, we try to verify greater than comparison,

GT (Greater than) = 1 if $A > B$.

So, the condition for $A > B$ or $GT = 1$ if and only if:

- $A_3 > B_3 \Rightarrow A_3 = 1$ and $B_3 = 0$, or
- $A_3 = B_3$ and $A_2 > B_2$ or
- $A_3 = B_3$ and $A_2 = B_2$ and $A_1 > B_1$, or
- $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0 > B_0$

Therefore,

$$GT = A_3 B_3' + X_3 A_2 B_2' + X_3 X_2 A_1 B_1' + X_3 X_2 X_1 A_0 B_0' \quad \text{——— (ii)}$$

X_3 is written when $A_3 = B_3$ as derived in equality. Similarly X_2 and X_1 are written.

For less than comparison,

$$LT = A_3' B_3 + X_3 A_2' B_2 + X_3 X_2 A_1' B_1 + X_3 X_2 X_1 A_0' B_0 \quad \text{——— (iii)}$$

Circuit Diagram:

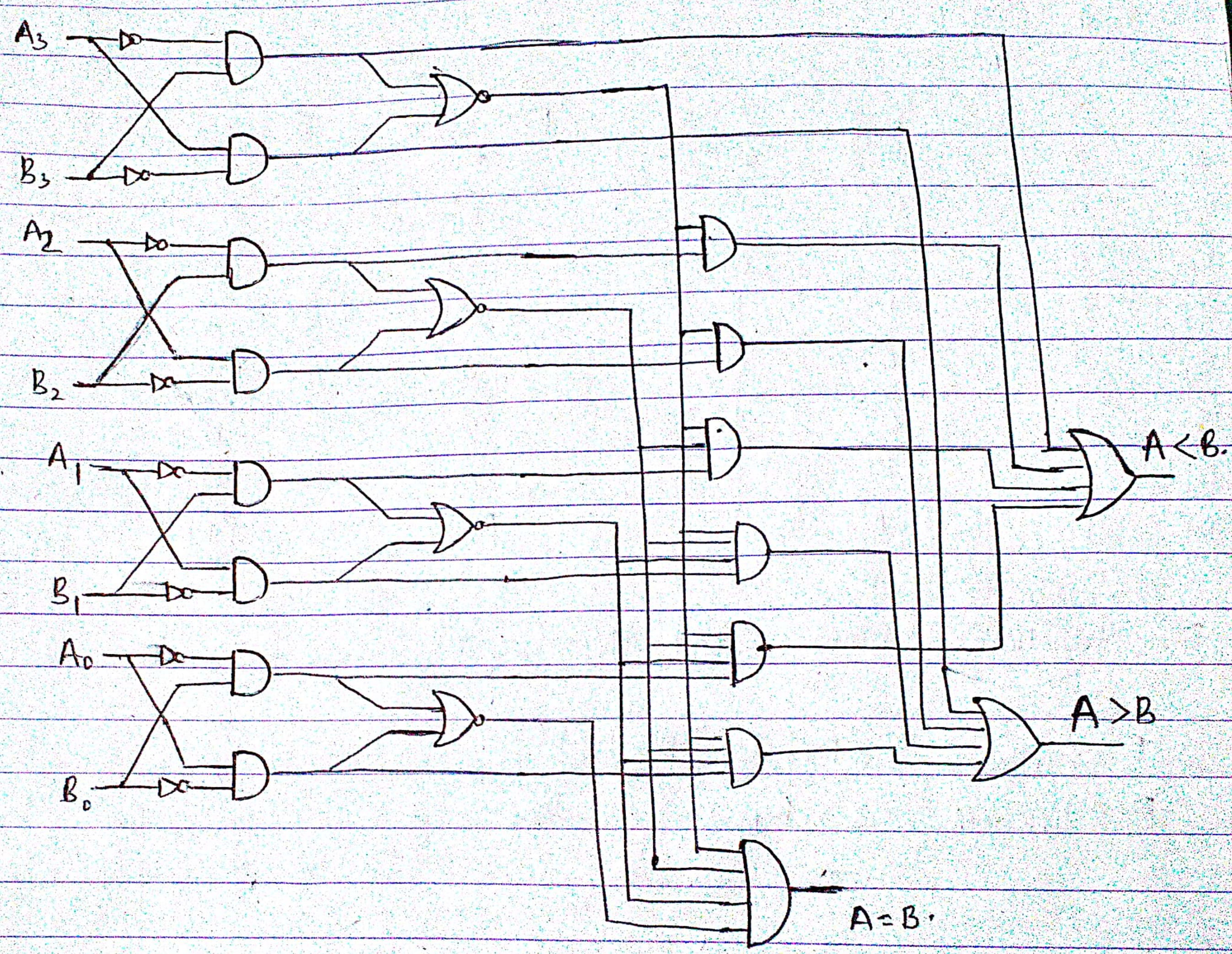


fig. 4-bit Magnitude comparator.

3. Explain full subtractor using decoder.

→

It is a combinational circuit that performs a subtraction betⁿ two bits taking into account that 1 may have been borrowed by a lower significant stage. The circuit has ~~been~~ three inputs and

two outputs. The 1's and 0's for output variables are determined from subtraction of $(x-y)-z$.

Truth Table:

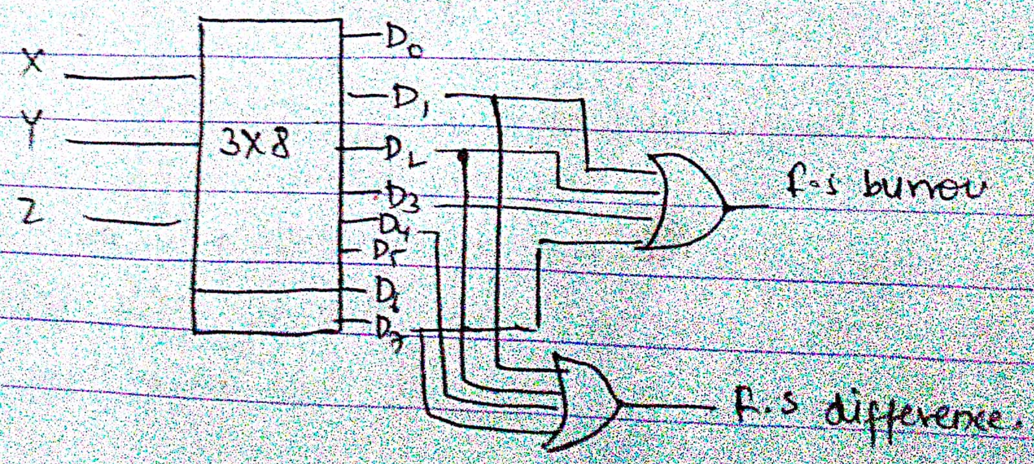
	Inputs			output	
	X	Y	Z	F.S burrow	F.S difference
D_0	0	0	0	0	0
D_1	0	0	1	$\boxed{1}$	$\boxed{1}$
D_2	0	1	0	$\boxed{1}$	$\boxed{1}$
D_3	0	1	1	$\boxed{1}$	0
D_4	1	0	0	0	$\boxed{1}$
D_5	1	0	1	0	0
D_6	1	1	0	0	0
D_7	1	1	1	$\boxed{1}$	$\boxed{1}$

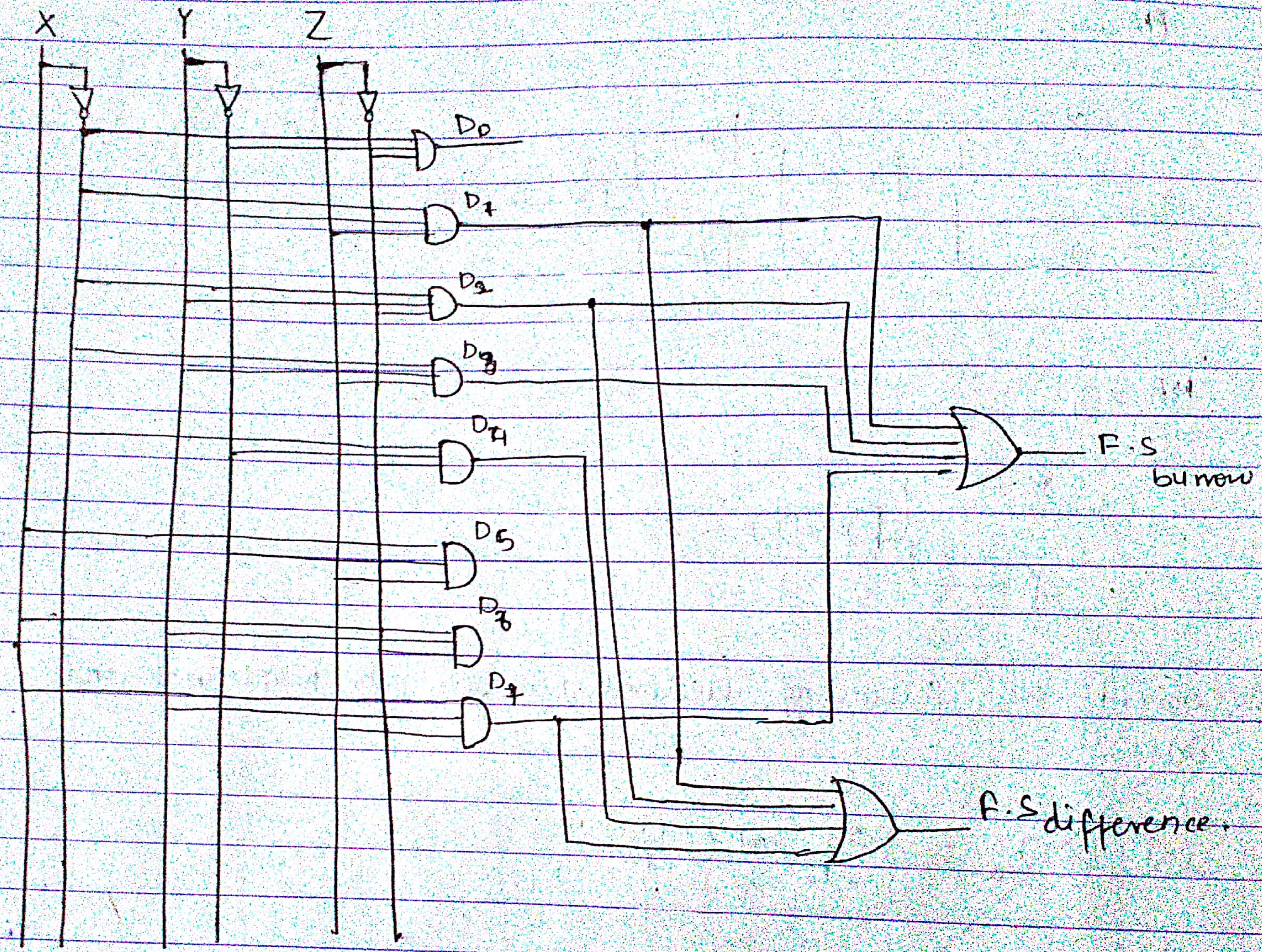
The boolean of function burrow and difference is:

$$F.S_{\text{burrow}} = \Sigma(D_1, D_2, D_3, D_7)$$

$$F.S_{\text{difference}} = \Sigma(D_1, D_2, D_4, D_7)$$

Since, there are three inputs and outputs, we can implem 3x8 decoders.





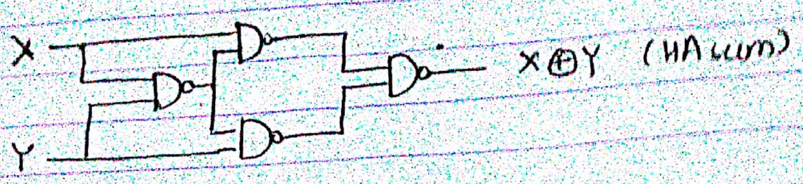
4. Design Half adder logic only using NAND gates.

→ We have,

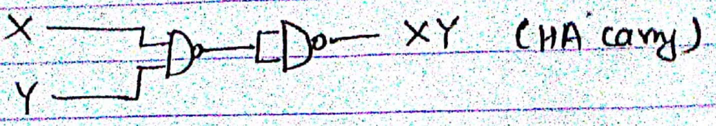
$$HA_{sum} = xy' + x'y = x \oplus y$$

$$HA_{carry} = xy$$

HA sum:



HA carry:



5. Convert the following decimal numbers into hexadecimal and octal.

(a) 334.

(i) $(334)_{10} \rightarrow (?)_{16}$.

$$
\begin{array}{r|l}
16 & 334 \rightarrow 14 \\
16 & 20 \rightarrow 4 \\
 & 1 \rightarrow 1
\end{array}
$$

$\therefore (334)_{10} \Rightarrow (14E)_{16}$

(ii) $(334)_{10} \rightarrow (?)_8$

$$
\begin{array}{r|l}
8 & 334 \rightarrow 6 \\
8 & 41 \rightarrow 1 \\
 & 5 \rightarrow 5
\end{array}
$$

$\therefore (334)_{10} \Rightarrow (516)_8$

b) 225

(i) $(225)_{10} \rightarrow (?)_{16}$

$$
\begin{array}{r|l}
16 & \underline{225} \rightarrow 1 \\
 & 14 \rightarrow 14
\end{array}
$$

$\therefore (225)_{10} \Rightarrow (E1)_{16}$

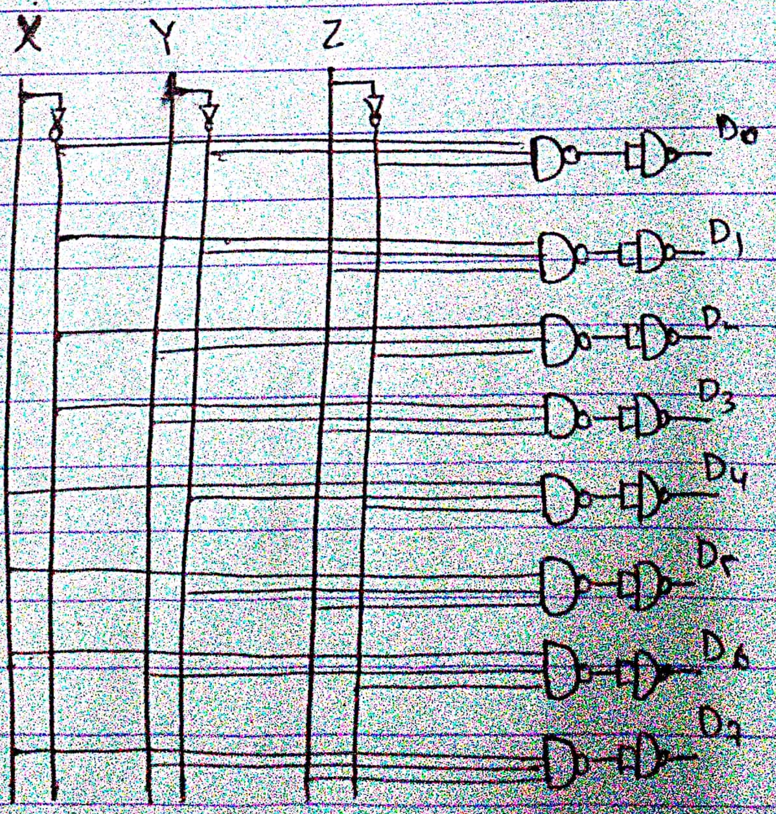
(ii) $(225)_{10} \rightarrow (?)_8$

$$
\begin{array}{r|l}
8 & \underline{225} \rightarrow 1 \\
8 & \underline{28} \rightarrow 4 \\
 & 3 \rightarrow 3
\end{array}
$$

$\therefore (225)_{10} \Rightarrow (341)_8$

11. Explain and design decoder with universal gates.

→ for 3×8 decoder,

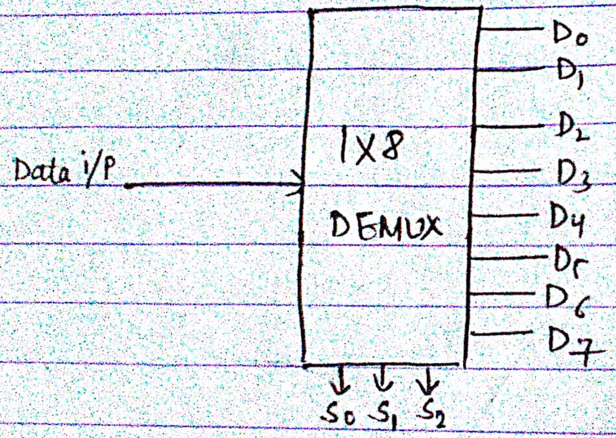


– Gaurav Chaulagain
– Digital logic

2071.

3. What is demultiplexer? Draw its block diagram and explain its working principle.

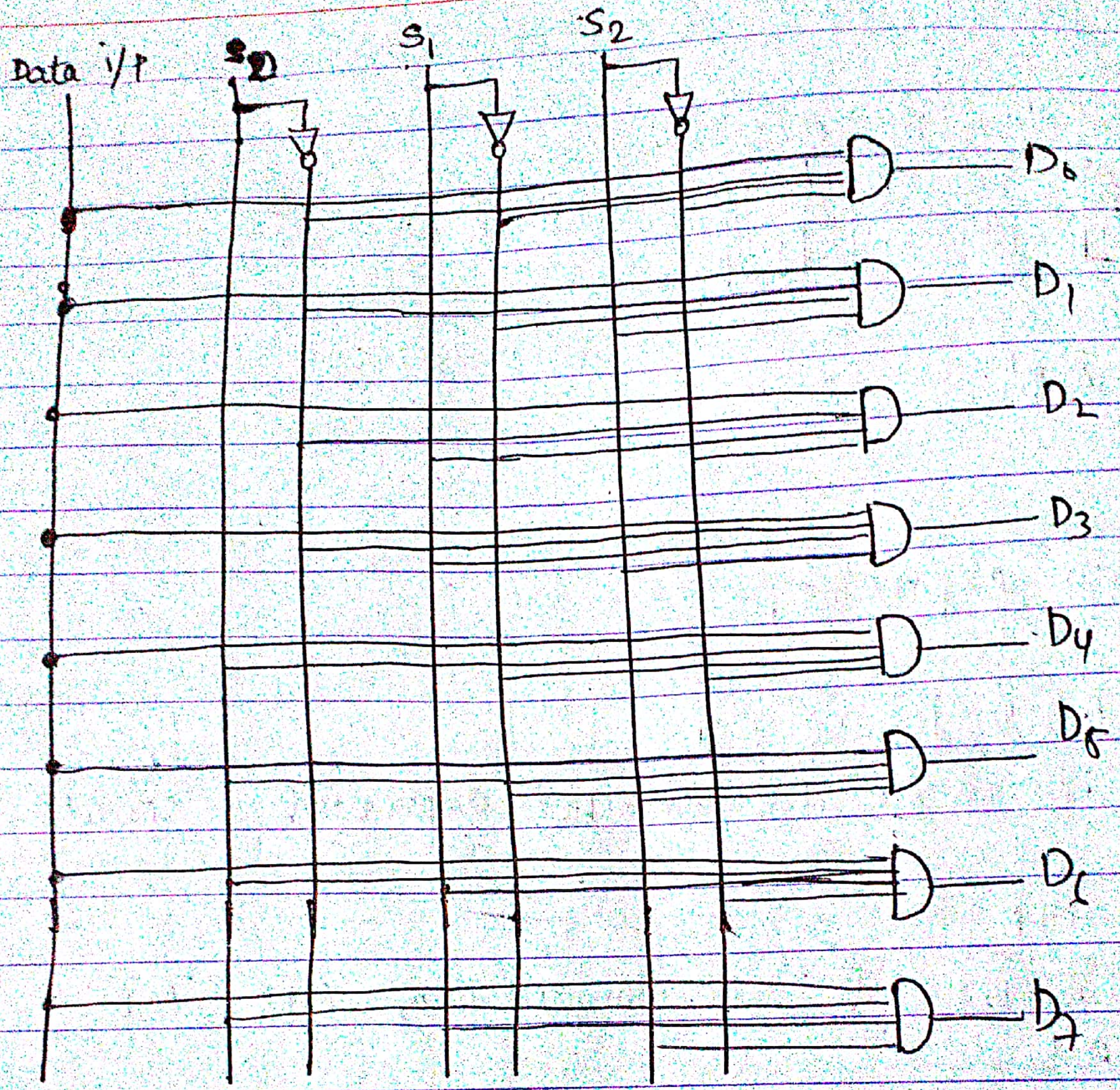
→ Demultiplexer is a data distributor which reverses the multiplexer function by taking digital information from 1 line & distributing it to a given number of output lines. It recieves information on a single line and transmitted information are of 2^n possible output lines. The selection of a specific output lines is controlled by the bit value of selection switches



Truth table:

output

Data I/P	S_0	S_1	S_2	~~output~~	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
D_0	0	0	0		1	0	0	0	0	0	0	0
D_1	0	0	1		0	1	0	0	0	0	0	0
D_2	0	1	0		0	0	1	0	0	0	0	0
D_3	0	1	1		0	0	0	1	0	0	0	0
D_4	1	0	0		0	0	0	0	1	0	0	0
D_5	1	0	1		0	0	0	0	0	1	0	0
D_6	1	1	0		0	0	0	0	0	0	1	0
D_7	1	1	1		0	0	0	0	0	0	0	1



simplify the boolean function using 3-variable k-Map.

(a) $F(X, Y, Z) = \Sigma(0, 3, 2, 5)$

X \ YZ	Y'Z'	Y'Z	YZ	YZ'
X'	1		1	1
X		1		

$F = X'Z' + Y + XY'Z$

(b) $F(A,B,C) = \Sigma(0,2,4,5,6)$.

A \ BC	B'C'	B'C	BC	BC'
A'	1			1
A	1	1		1

$F = C' + AB'$

7. simplify the boolean expresion:

$$Y = \overline{A \cdot B} + \overline{\bar{A} + \bar{B}}$$

prepare truth table to show that simplified expresions is correct or Not

→

$$Y = \overline{A \cdot B} + \overline{\bar{A} + \bar{B}}$$
$$= \overline{A \cdot B} + (\bar{\bar{A}} \cdot \bar{\bar{B}}) \quad [\because \text{De morgan's law, } \overline{A+B} = \bar{A} \cdot \bar{B}.$$
$$A + A' = 0.$$
$$= \overline{A \cdot B} + A \cdot B$$
$$= 1 \qquad [\because A + A' = 1]$$

Truth table.

A	B	$\bar{A}$	$\bar{B}$	$A \cdot B$	$\overline{A \cdot B}$	$\bar{A} + \bar{B}$	$\overline{\bar{A} + \bar{B}}$	$\overline{A \cdot B} + \overline{\bar{A} + \bar{B}}$
0	0	1	1	0	1	1	0	1
0	1	1	0	0	1	1	0	1
1	0	0	1	0	1	1	0	1
1	1	0	0	1	0	0	1	1.

Hence, verified

Scanned by CamScanner

Q. Explain the PLA (Programmable logic Array).

→ A combinational circuit may have Don't care conditions and when implemented with a ROM, a Don't care condition becomes an address input that will never occur. The words at the don't care address need not be programmed and maybe left in the original state. The result is that, not all the bit patterns availab in the ROM are used, which maybe considered a waste of availab equipments. Thus, for cases where no. of Don't care conditions are excessive, it is more economical to use a component called programmable logic array (PLA).

A PLA is similar to RAM in concept. However, the PLA does not provide full decoding of the variables and does not generate all the minterms as in the ROM. In PLA, the decoder is replaced by a group of AND gates, each of which can be programmed to generated a product term of the input variable.

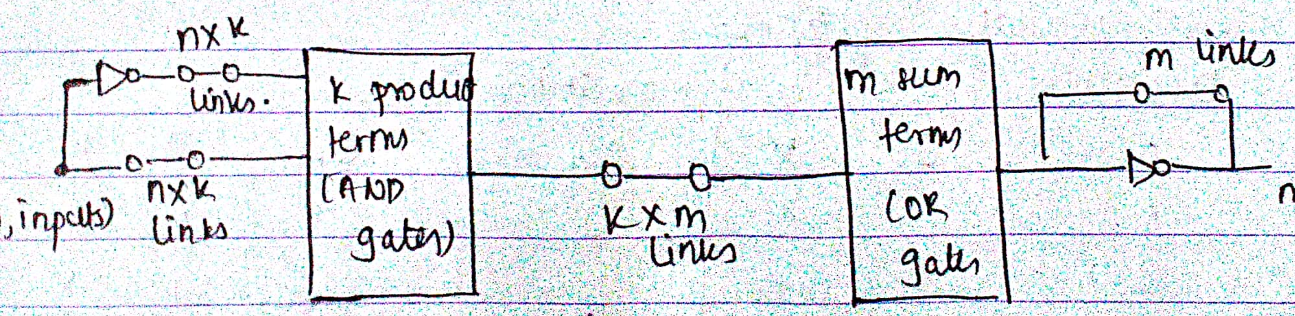


fig. PLA block diagram.

9. How JK flipflop can be converted into a D-flipflop? Explain.

→ steps:

1. Identify available and required flip flop.
2. Make characteristic table for required flip flop.
3. Make excitation table for available flip flop.
4. Write boolean expression for available ff.
5. Draw the circuit.

Here,

available flipflop = J K flipflop

required flipflop = D flipflop

Characteristic Table for D flip flop:

Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1.

Excitation table for available flipflop:

We know,

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation table:

Q_n	D	Q_{n+1}	J	K
0	0	0	0	X
0	1	1	1	X
1	0	0	X	[illegible]
1	1	1	X	[illegible]

Boolean expression:

for J,

Q_t \ D	D'	D
Q'	0	1
Q_t	X	X

$J = D$

for K,

Q_t \ D	D'	D
Q_t'	X	X
Q_t	1	0

$K = D'$

Circuit:

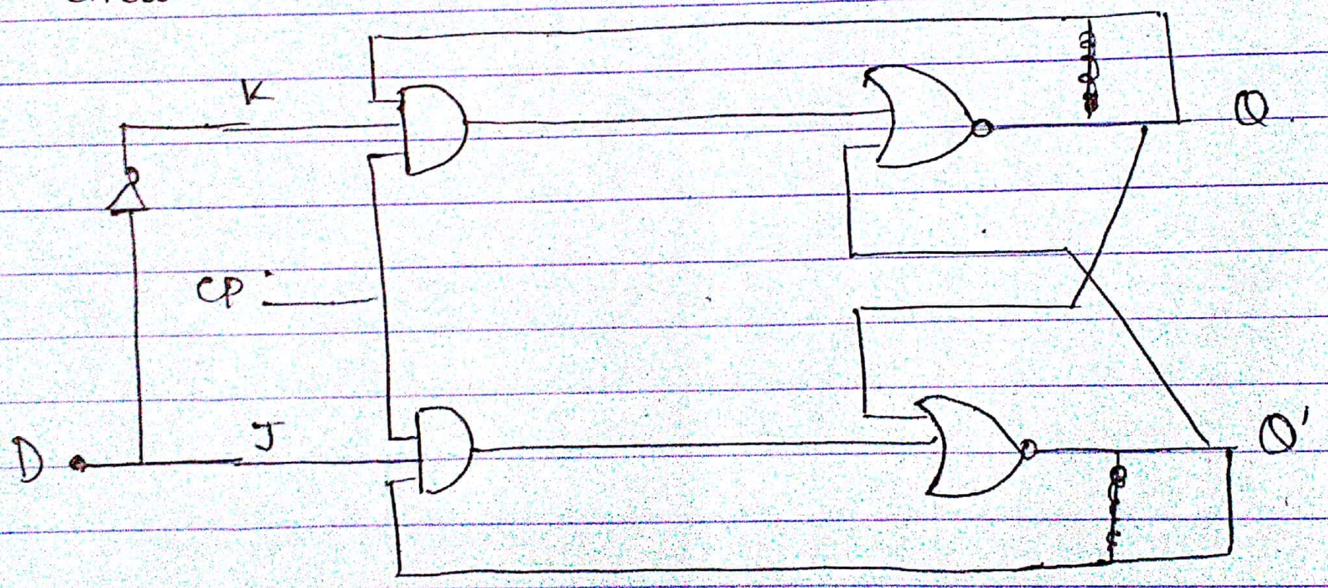


fig: D-flipflop using J-K flipflop.